ECE2277A Lab 2: Seven-Segment Display Driver

Professors Arash Reyhani & John McLeod
Prepared by John McLeod

2022 10 17 to 2022 10 28

Objectives

This lab is an exercise in designing a circuit to convert a binary input to a human-readable output. The binary input will be provided by sliding switches, the human-readable output will be provided by a seven-segment display panel (7SD). The objectives of this lab are:

1. To understand how to construct circuits that display human-readable output on a 7SD,
2. To learn how to use more complicated output hardware on the FPGA board, and
3. To demonstrate the functionality of the hardware implementation of that logic circuit.

This lab will be done using Quartus, employing the techniques described in Lab 0 and Lab 1. As with Lab 1, extra steps are required to program the FPGA development board.

Summary

A summary of the deliverables for this lab are enumerated below.

1. Derive a Boolean expression to detect when an 4-bit code does not correspond to a valid BCD value.
2. Design a circuit in Quartus to implement this expression, and also display the 4-bit code on a 7SD.
3. Program this circuit onto the DE10-Standard FPGA development board.
4. Demonstrate the correctness of your design to the TA by toggling switches and observing what is displayed on the FGPA board.

You should derive the required Boolean expression *before* coming to the lab. Ideally, you should also implement the Quartus design *before* coming to the lab, that way you can focus on programming the FPGA board and demonstrating your work to the TA.

Grading Scheme

The lab is graded as follows, out of a total of 50.

Section	Details	Mechanism	Grade
Expression Derivation	Minimized Expression	Show written work to TA	10
Quartus Circuits	Construction	Show Quartus project to TA	10
FPGA Hardware	Implementation	Demonstrate working hardware to TA	20
Concepts	Answer Questions	Discuss exercise with TA	10

Please bring in evidence of how you derived the Boolean expression (i.e. a K-map or lines of Boolean algebra) to show the TA. This can be handwritten, printed, on a laptop or tablet, etc.

Quartus Tips

The laboratory computers in ACEB2400 use Quartus 18.0, while most of you probably have Quartus 20.1. There are (probably) some backwards-compatibility issues between these two versions, so a finished Quartus 20.1 project (probably) won't work in Quartus 18.0. However for these simple lab exercises you don't need the entire project — just the block-diagram schematic file (.BDF) and the pin assignment file (.QSF).

- You can design the logic circuit in Quartus at home (or in the general computer labs), and compile it to make sure there are no errors, *before* your lab session.
- You can bring the completed .BDF file (on Western OneDrive, a USB stick, or whatever) to the lab.
- In the lab, start a new, blank, project with the same name as the one you created previously.
- Transfer the .BDF file into the empty project directory, and open it in your project (*File→Open*).
- Assuming all file names are consistent, and the circuit was correctly designed, you should be able to compile this project in the lab without any issues.

Furthermore, because we need to program the actual FPGA hardware, the pin assignments for input/output are crucial.

- Because the 7SD has, well, seven outputs, there are a lot of pins to assign.
- You can carefully make all the pin assignments in your Quartus project at home using the list of pin assignments provided in the lab.
- Once complete, you can export the assignments (*Assignments→Export Assignments...*) to create a .QSF file. I suggest giving this .QSF a **new** name that is not the same as the project.
- You can bring the .QSF file (on Western OneDrive, a USB stick, or whatever) to the lab as well.
- In the lab, transfer the .QSF file into the new project directory, and import it into your project (*Assignments→Import Assignments...*, and find the appropriate file).
- Assuming all file names are consistent, and the circuit was correctly designed, you should be able to recompile this project in the lab with the completed pin assignments without any issues.

You don't have to do these steps before coming to the lab, but it is strongly advised. This will help you make more efficient use of the laboratory time fighting with Quartus to program the FPGA board, instead of fighting with Quartus to design a circuit.

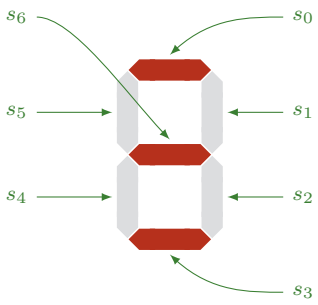
Seven-Segment Displays

This is the first lab exercise that will make use of a 7SD panel. All of you are probably familiar with this piece of hardware: it is a small panel with 7 light-emitting strips that can be illuminated to represent decimal, octal, or hexadecimal digits, as well as some other letters and symbols. A non-programmable calculator typically uses seven-segment displays, assuming kids these days still know what a non-programmable calculator is. This panel also actually includes a light-emitting dot to act as a radix point, so perhaps they really are "eight-segment displays". But, as far as I can tell, the radix point on these displays is not wired to the FPGA board, so with the hardware available only the 7 segments can be controlled.

A 7SD is a great piece of hardware for combinational circuit design, and we will use these in lab 4 as well as the project.

- A digital system will use binary or BCD coded values, which are not very human-readable.
- A 7SD can show numbers using the Arabic numeral system we are all familiar with.
- However, *between* the digital system and the 7SD, a completely different kind of code is needed.

A 7SD allows each light-emitting segment to be toggled on or off individually, therefore it accepts 7 inputs $S = s_6 s_5 s_4 s_3 s_2 s_1 s_0$. A 7SD will show a unique pattern for 7-bit binary number used as an input, regardless of whether that corresponds to the appropriate symbol for an Arabic numeral or a Roman letter. The input s_0 controls the top-most segment, remaining inputs control segments proceeding clockwise around the panel to s_5, the segment in the centre is controlled by s_6.



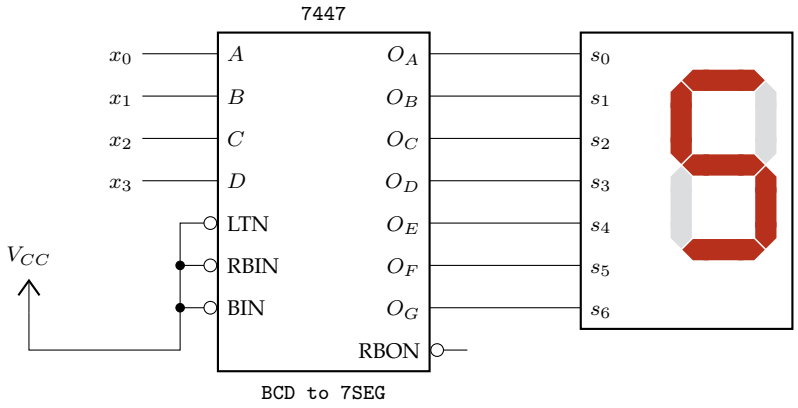
To operate a 7SD, we need to design a combinational circuit that accepts a 4-bit input (corresponding to a hexadecimal or BCD digit), and generates the appropriate 7-bit output pattern to display that digit on the 7SD.

- The 7SD panels on the Terasic DE10-Standard development boards are active-low, so an input $S =$ `0b1111111` will blank out the panel, and an input $S =$ `0b0000000` will illuminate every segment (displaying the digit "8").

To design the combinational circuit to control a 7SD, you would need to determine the appropriate output patterns for each digit. As mentioned above, `0b0000000` is "8", `0b0011001` is "4", etc. From this, you would draw a K-maps for each output.

Fortunately for you, this design process is not necessary. The symbol library in Quartus already contains an implementation of a BCD-to-seven segment display driver, called a `7447` chip. You can find this in the symbol placement dialog (same one you use to find `AND` and `OR` gates) under `others`→`maxplus2`→`7447`.

- The four variable inputs are labelled `A`, `B`, `C`, `D`. Please note that `A` is the **least significant bit** and `D` is the **most significant bit**: a binary number $x_3x_2x_1x_0$ should be implemented using the variables $DCBA$.
- The seven 7SD outputs are labelled `OA`, `OB`,... through `OG`. Again, `OA` is the **least significant bit** and `OG` is the **most significant bit**. Make sure you follow the pin assignment table given below in *Lab Procedure*, otherwise you may mis-wire the logic outputs to the 7SD hardware and the wrong symbols may be displayed.
- Three control inputs for the `7447` chip — `LTN`, `RBIN`, and `BIN` — should be set to a constant input of logical 1. This is implemented as `VCC` in Quartus. In the symbol library, look for `primitives`→`other`→`vcc`.
- A fourth control input, `RBON`, can be left hanging. Do not connect anything to this pin.



A circuit using the `7447` block in Quartus is shown above.

Lab Preparation

To prepare for the in-person lab exercise, derive the logic circuit that determines whether or not a 4-bit code is a valid BCD value.

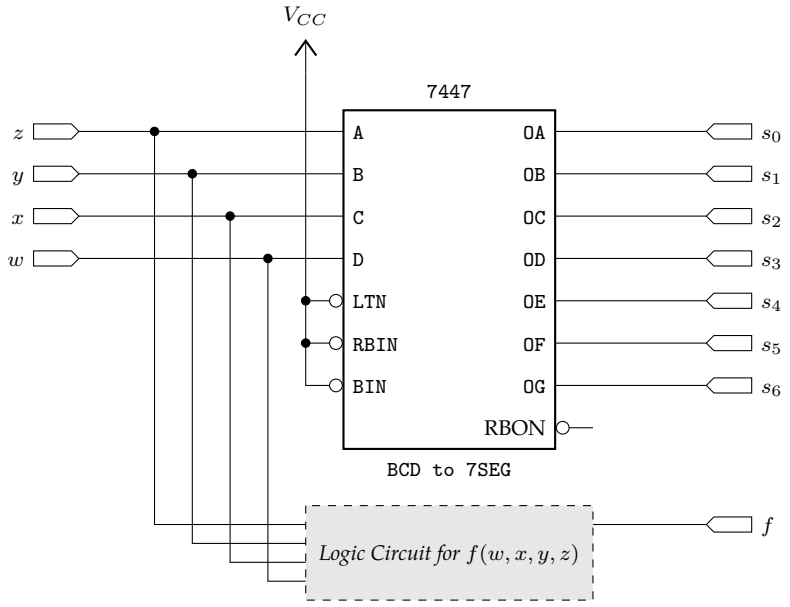
1. The 4-bit input is assigned the variables w, x, y, z, where w is the most significant and z is the least significant bits. Therefore, $(w, x, y, z) = (1, 0, 0, 0)$ is 8_{BCD}, and $(w, x, y, z) = (0, 0, 0, 1)$ is 1_{BCD}.
2. The logic output is $f(w, x, y, z) = 1$ if the sequence (w, x, y, z) is a **invalid** BCD value, and $f(w, x, y, z) = 0$ if the sequence (w, x, y, z) is a **valid** BCD value.

This logic circuit output $f(w, x, y, z)$ will be assigned to an LED on the FPGA board, so that the LED lights up when an invalid BCD value is given.

You **must** complete this circuit derivation **before** coming to the lab — in order to start the lab you must show the TA your completed derivation.

Lab Procedure

This lab emphasizes programming a FGPA to implement a logic circuit in hardware. This lab will use four slider switches on the DE10-Standard board as the inputs. The output will be one 7SD and one LED. Consequently, there are four input pins and eight output pins.



1. Design **circuit 1** in Quartus (make sure you use the `5CSXFC6D6F31C6`, for the DE10-Standard, as the device when creating the project), using the appropriate logic circuit obtained in the prelab for $f(w, x, y, z)$.
2. The inputs w, x, y, z should be controlled by slider switches, so make sure the pin assignments follows those given in the table below.

Algebraic Name	Hardware Label on FGPA Board	Pin Name in Quartus
w	`SW3`	PIN_AC30
x	`SW2`	PIN_AB28
y	`SW1`	PIN_Y27
z	`SW0`	PIN_AB30

If you want to use different switches for some reason that is fine (there are 10 switches available), just make sure you remember which switch is associated with which input variable. A complete list of the pin names for all switches is available on OWL.

3. The output $f(w, x, y, z)$ should be an LED, so make sure the pin assignment follows that given in the table below.

Algebraic Name	Hardware Label on FGPA Board	Pin Name in Quartus
$f(w, x, y, z)$	`LEDR0`	PIN_AA24

If you want to use a different LED for some reason that is fine (there are 10 LEDs available), just make sure you note which one you use. A complete list of the pin names for all LEDs is available on OWL.

4. Finally, the outputs $O_G(w, x, y, z)$ to $O_A(w, x, y, z)$ should correspond to a 7SD, so make sure the pin assignment follows that given in the table below.

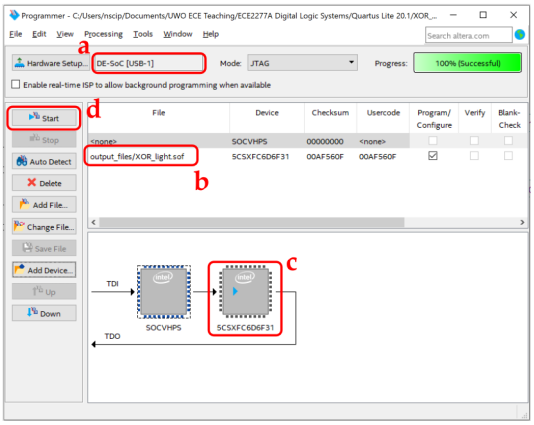
Algebraic Name	Hardware Label on FGPA Board	Pin Name in Quartus
$s_0(w, x, y, z)$	`HEX0`	PIN_W17
$s_1(w, x, y, z)$	`HEX0`	PIN_V18
$s_2(w, x, y, z)$	`HEX0`	PIN_AG17
$s_3(w, x, y, z)$	`HEX0`	PIN_AG16
$s_4(w, x, y, z)$	`HEX0`	PIN_AH17
$s_5(w, x, y, z)$	`HEX0`	PIN_AG18
$s_6(w, x, y, z)$	`HEX0`	PIN_AH18

If you want to use a different 7SD for some reason that is fine (there are six 7SD panels available), just make sure you note which one you use. A complete list of the pin names for all 7SDs is available on OWL.

5. Compile the project in Quartus and make sure there are no errors.

It is possible to complete all of the above steps on any computer with Quartus before coming to the lab. Note that it is not necessary to simulate the operation of this circuit, however you are welcome to do the simulations if you want to test your design before bringing it to the lab.

6. Make sure "DE-SoC (USB 1)" is selected near the *Hardware Setup* button (**a**). If it is not, press the button and try to find it.



7. Program the FPGA board using Quartus (*Tools→Programmer*).
   - Make sure a icon representing the ARM Cortex-A9 processor ("hard processor system", or HPS) is visible. It should be called `SOCVHPS`, and it should be before the `5CSXFC6D6F31` icon representing the FPGA in the bottom part of the dialog window. If it is not visible, click the *Add Device...* button, and find *SoC Series V→*`SOCVHPS` in the popup window. (Alternatively, you can mess around with *Auto Detect* button, as I do in the video, but this way is probably more straightforward.)
   - Make sure the appropriate .SOF file is present in the middle part of the dialog window (**b**). If it is not, click the *Add File* button and find it. It should be somewhere in the project directory.
   - Make sure the `Program/Configure` box next to the .SOF file name is checked. If it is not, check it.
   - Make sure the `5CSXFC6D6F31` icon representing the FPGA is selected. It should have a blue triangle on it (**c**). It it does not, click it.

- If everything is done correctly, the *Start* button should be available (**d**). Click it to start programming. The progress bar will turn green and read *100% Successful* if it worked.

8. Once programming is complete, you can toggle the slider switches on the FPGA board and see how that changes the 7SD and LED output.

Once your FPGA board is programmed and working correctly, call the TA over to demonstrate your work. There are also a few conceptual questions the TAs may ask you.